Parallel Training for Back Propagation in Character Recognition

Mumtazimah Mohamad, Md Yazid Mohd Saman, Muhammad Suzuri Hitam
Department of Computer Science
University Malaysia of Terengganu
Kuala Terengganu, Terengganu, Malaysia
ummurifqi09@gmail.com, yazid, suzuri {umt.edu.my}

Abstract **— Artificial Neural Network has made the character recognition work easier and they grow tremendously in improving accuracies and efficiency. However, there are always gaps and weaknesses which is need to prevail due to recognition inaccuracies and less discussed especially in handling large scale of data. The parallelism can be regarded as the practical solution in solving large workload. In achieving an optimal training time and generalization ability, possessing the problem for generating suitable comprehensive classifier will affected positively to the time and maintaining the accuracy in the same time. This paper presents an idea of distributing data the same neural network structure to measure the capability of time reduced for recognition accuracy. The MNIST benchmark dataset is used represent handwritten digits for recognition. To test the validity, the data sets are handled in parallel computers in separate training of back propagation neural network. The results indicate the proposed algorithm improve the speed- up performance for large scale neural networks while maintaining its accuracies. (Abstract)**

Keywords-component; MNIST Data Set; Parallel neural network; distributed memory, neural networks (key words)

I. INTRODUCTION

Character recognition has an importance role in various fields such as traffic tracing, valuable script discrimination and others which has been extremely studied [1-4]. Character recognition is the mostly area where artificial neural network (ANN) techniques made their first significant contribution to computer science application and proved to be reliable alternatives to classical methods. The multilayer feed forward networks comprises input layer, one or more hidden layers of nodes, and an output layer of nodes. The input layer nodes propagate some computation in a forward direction, on a layer by-layer basis. The back-propagation (BP) algorithm is a popular incremental learning where it has been proven as most universal approximation technique by Rumelhart(1986) [5, 6]. This algorithm has been considered as delta rule generalization for non linear function.

This paper will focus on how large data can be handled in BP where, there are still remaining problem of computationally expensive especially in time spent [6, 7]. Other problem of BP such as iteration computing for good performance [8] and less accuracy and accepted error rate in training [1, 8, 9]. One of the practical solutions is to exploit the basic idea of ANN by parallelizing its concurrency. Expensive computation of BP are distributed over set of computers for computational purpose [10]. This paper will describe the handwritten character recognition using ANN cluster to improve the learning time while preserving the accuracy constructed large scale of ANN.

II. LEARNING METHODS FOR CHARACTER RECOGNITION

ANN learning methods are enormously utilized for character recognition and make the workload easier especially in improving accuracies. There are a lot of excellent results that proved the advantages of neural network [7, 9, 11]. However, there are a few opportunity of improvement for ANN in managing large dataset for handwriting recognition either machine printed character or degraded image. There are a few research focuses large dataset because ANN consume a higher computational cost [6]. In addition, majority of previous and current researches focuses solution on its feature extraction recognition and not to original image pixels as a whole. It is understood that the solution should rely on better learning utilizing to the samples. This paper will mainly discuss on the statistical methods which is focusing in the supervised ANN.

The most basic way of recognizing patterns is the probabilistic methods. This has been done by using the Bayesian decision theory as mentioned in [6]. The popular statistical as well as structural approaches technique is back propagation neural network [1, 3, 8, 12]. The character recognition also best discussed in Support Vector Machine [13, 14] and also in Convolutional Network [6, 11]. There are others ANN technique such as RProp training [15], QuickProp [16], conjugate gradient [13], recurrent network [14], Hopfield network[17] and Self Organizing-Map [18]. This work focused on back propagation method due to its efficiency of learning with the strong and consistent theory behind it.

III. BACK PROPAGATION ARTIFICIAL NEURAL NETWORK

The back propagation neural network (BPNN) is a gradient descent algorithm by minimizing the error signal between actual and target output of multilayer perceptron [19]. BPNN solves compounding errors between interacting modules by back propagating error information throughout a network [20]. The BP comprises a forward propagate phase, back propagate phase and weight update phase. Figure 1shows chain rule represented by arrows, involves the weights of output and the hidden layer where the initial weight will be established to derived error function. The network adapt and generalize [10, 21, 22] themselves easily with the data by repeating the three phases until the error signal achieve its target. These steps capable of learning complex input to output relationships by sequential training procedures. Equation (1) shows the first phase which is to get the network response after representing pattern data into the network. This feed forward phase can be formulated based on Werbos's (1974) as follows:

$$y = F_3\left(\sum_{j=1}^{N} W_{j3}\left(F_2\left(\sum_{i=1}^{N} W_{ij}x_i - T_j\right)\right) - T\right) \quad (1)$$

In (1), N is the number of nodes of corresponding error while wj3 is the weight of node j of the hidden layer to the nodes. The wij and xi are the weight for input node to the node of hidden layer and the output values. Tj is the threshold for the output nodes. The error signal is based on the weights of the neural network [5, 23]as in (2). The error measures in sum square error for the different of the output y compared to the desired output t.

$$E = \frac{1}{2}\sum_{p=1}^{p}(t - y)^2 \quad (2)$$

The error is then minimized by calculating the gradient descent with corresponding weight for each node of layers. The weight update starts from output layer to the hidden layer by propagating the error to each layer. The update phase based on follows [24]:

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{p} = \left(-\eta\frac{\partial E_p}{\partial w_{ij}}\right) = \sum_{p=1}^{p} \Delta_p w_{ij} = \sum_{p=1}^{p} \Delta_p w_{ij} = \eta \sum_{p=1}^{p} \delta_{pi} O_{ij} \quad (3)$$

Equation (3) represents how the weight and learning gradient $\Delta_p w_{ij}$ is calculated in each cycle. δ_{pi} represent the error term of each layer and η represent the learning rate. The activation function of the hidden layer nodes should be a non linear function to linearly separable problems, which is the best utilized by of BPNN. In this case, sigmoid function is used because it is easy to determine whether the network is good or rich enough for learning. The changes of the weight will affect the output even for smaller changes.

A. Steps in BPNN

- The MNIST dataset are randomly distributed from all digits to get the generalized results.
- The dataset already comprises into two parts; the training (70%) and the testing datasets (30%).
- Training of BPNN using learning rate and momentum.
- The testing of the dataset for unseen pattern for classification accuracy

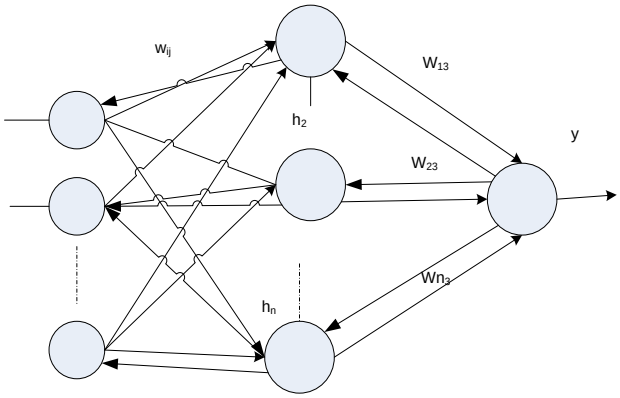

Figure 1. Input unit, hidden unit and output unit of BPNN

IV. PARALLEL BACK PROPAGATION

A. Introduction to Parallel Processing

Parallel processing is a collection of independent and interconnected computers processing that capable of collaborating on a task concurrently. The single processor computation should be faster by using a number of processors with time and efficiency of any given ANN [[9, 10, 25]. The pyramidal structure of BPNN Learning can be utilized together with the parallel processing advantage. The parallel learning has an important role where the size of dataset is expected to be increased considerably faster than computational capabilities [26]. Distributing training part to more than one processors enable large scale computational, economic and reliable. The parallelism can manage long training time in sequential ANNs within the minimal time. The parallel model in this paper used based on single instruction, multiple data (SIMD) with distributed memory. All processing unit execute the same instruction at any given clock cycle. This architecture needs a communication network to connect between the available processors.

Previous researches done in improving the training and usage has led to the development to an increased training. Many works focused on special purpose hardware implementations that facilitates high level; parallelism, according to the architecture of visual share memory [10], cluster system [26] and on networks of stations [5]. Several approaches are on the algorithm where mostly they use Message Passing Interface (MPI) in providing parallelism over processors [24, 27, 28]. There following strategies that

has been proposed for parallelizing the back propagation algorithm of MLP:

- Training session parallelism: Each processor run with different initial training parameter where it can determine which training parameter is suitable most [29]
- Network partitioning: Each unit of neural network layer is distributed among processors [30]
- Pattern Partitioning: Each processor runs a whole neural network locally for an assigned and partitioned dataset pattern by master. Each processor hold partial gradient before synchronize it with master unit [22, 31]and
- Combination of network partitioning and pattern partitioning [32]

The batch processing used because it helps faster convergence in large dataset compared to online learning in generalizing the optimal weight [33]. The learning rate used is between 0 and 1 in order to ensure the convergence of the optimal weight. The parallel training involves a process of partitioning the training session which is straight forward based on sequential back propagation algorithm [30]. Each processor has the whole copy network. Each of them presents the different pattern block for each training cycle as in (3). Component gradient is the result of individual processor that represents some part of the batch of the training pattern as summarized in (4) and (5). After each batch pattern presented, then master will synchronize the component gradient as follows:

$$\bar{g}^{<q>} = \sum_{p \in B_q} \frac{\partial E_p}{\partial \bar{w}} \qquad (4)$$

where $\bar{w}$ represents the set of all weights. The $\Delta B^{\bar{w}}$ to the representing weight changes in batch B to the corresponding sum of component gradient $\bar{g}^{<q>}$:

$$\Delta B^{\bar{w}}(n+1) = \eta \sum_{p \in B_q} \frac{\partial E_p}{\partial \bar{w}} + \alpha \Delta B^{\bar{w}}(n)$$

$$= \eta \sum_{q=0}^{p-1} \bar{g}^{<q>} + \alpha \Delta B^{\bar{w}}(n) \qquad (5)$$

Equation (5) describes the modification that involved only once in performing the collection and summation of all component gradient of $\bar{g}^{<q>}$ instead of each pattern in sequential algorithm. The component gradient in batch B communicated using neighbor configuration of P processors. Each P processor will send its component gradient and will receive p-1 processor's component gradient. Generally, all processor will receive temporary component gradient. Each processor implicitly exchanging the component gradient with the predecessor and receives a new component gradient. Each processor will received the complete component gradient from all processors. This means that each processor will able to update their weights by their own.

B. Parallelism using Message Passing Interface(MPI)

MPI is one of parallel standard application using message-passing paradigm for parallel processing. MPI provides a greater flexibility in managing and carrying parallel algorithm. It provides the optimized code for all parallel architectures and it is portable algorithm. During the processing, the set of tasks utilized local memory. A number of tasks can be carried out within the same physical machine and/or across arbitrary number of machines. MPI tasks are executed by a number of processors using MPI communication to communicate between them.

V. MNIST DATA SETS

The MNIST dataset is a standard handwritten digit classification and recognition benchmark. It is originally developed by the National Institute of Standards and Technology [34] and post-processed and distributed by Yann LeCun and Corinna Cortes. The original data, comprised of the NIST Special Database 1 (SD-1) and Special Database 2 (SD -3), represent hand written digits that have been digitized and post-processed to increase the available resolution and translate the centre of mass of each digit in increase the fidelity of data. The dataset contains the original black and white images. These images were normalized into a 20X20 pixel box while preserving their aspect ratio. To date, there are 60 000 training samples and 10 000 tests samples, with a resolution of 28 X 28 pixels and 256 bit-depth, that can be used to test and verify the effectiveness of various machine learning algorithm.

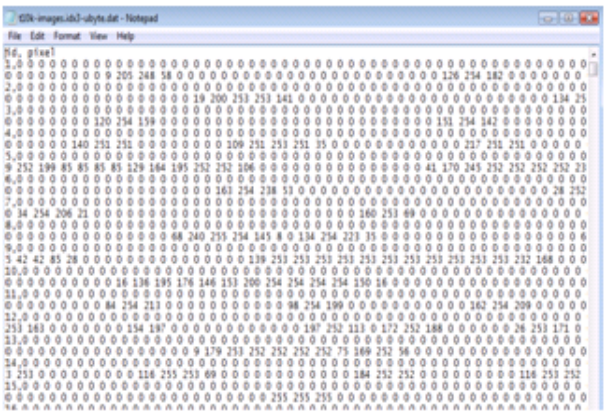

Figure 2. Extracted pixel in MNIST data sets.

The dataset is stored in vectors and multidimensional matrices. The integer data are stored in MSB first (high-endian) format used by most non-Intel processor. A simple program is used to read the dataset format which to flip the byte header in IDX file format of dataset. The image pixels are organized in row-wise in value 0 to 255. 0 represents background (white) while 255 means foreground (black).

VI. PARALLEL BACK PROPAGATION TRAINING

The parallelization techniques utilized a divide and conquer method. The training dataset T is partitioned into equal parts T1, T2, T3,..TN, and N is the number of parallel processors. The following algorithm in Fig.3 is parallel back propagation adapt from back propagation in (1), (2) and (3) in batch mode in (4) and (5).

Master start
1. Initialize the number of training dataset
2. Identify the processors for the parallel execution
3. Allocate the data in partitions for each available number of workers
4. Sends to the partitioned sub dataset to workers.
5. Workers:
   - 5.1. Calculate the value of sums of delta weights and delta thresholds.
   - 5.2. Calculate the worker's SSE for its assigned patterns
   - 5.3. Collective communication is executed (synchronization with other processors)
   - 5.4. Sum values of delta weights and thresholds transmitted to all working processors.
   - 5.5. The sum values of delta weights and thresholds j are then placed locally. Each processor uses these values to update the weights and the thresholds as point 5
   - 5.6. Update weights.
6. Master
   - 6.1. Determine the value of E(t) of the last cycle of the minimizing operation
   - 6.2. Determine whether to continue training or not.

Figure 3. Parallel back propagation

In this section, we use the back propagation with MPI Programming to network for 784 inputs, 30 hidden nodes, 10 output nodes and 784 input nodes with 1000 hidden nodes and 10 output nodes of three layer network to perform handwritten digit recognition. The back propagation algorithm is based on the cost function in equation (1) (2) and (3). The algorithm is simulated in the network with three different capacity learning rate: 0.05, 0.1, and 0.5. The cluster is set up to 8 nodes and using Intel Dual Core Machines. The algorithm is developed using C while the MPI [35] is implemented using MPICH2 and were compiled in Visual Studio 2005. The simulations were performed to validate the result by using different scenarios on the parallel performance. The following simulations were performed.

VII. RESULT AND DISCUSSION

Sequential and parallel training successfully developed and run in sequential and parallel algorithm. Figure 4 shows that speed up factor into 6 processors for 6 digit recognition. The greater processor shows the higher speed up factor in this case. There are a little bit increased due to global summation of parallelization. Each processor running exactly as sequential of a single processor and are independently from each other except for the exchange of delta of weights. The exchange of delta of weight is an extra work for parallel processor, thus it need an extra time. There are several hypotheses on parallel BPNN were answered. Firstly, the process of multiple processors can be measured as approximately like performance of the single processor.

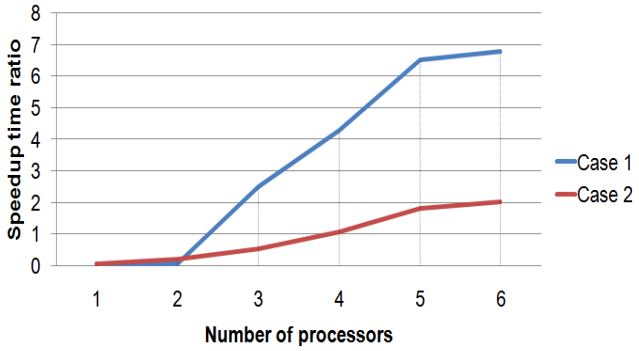

Figure 4. Speedup factor of parallel training

Secondly, the number of inputs in a given parallel training is larger from the sequential training which is performed better than sequential training. However, some performance degradation was expected due to MPI overhead. A sufficient large number of inputs in test data would decrease this effect, resulting in a performance gain. We have proved such experiment success for small system [36]]. Further test on larger distributes systems would help to accurately measure the systems' performance in more diverse environments.

Overall, batch training should be more efficient that online training for large datasets. The online training direct the gradient used for each parameter to a slow convergence [42]. However, in our case, batch learning slow down the convergence because of average weight changes during over each training cycle. Thus, in future these training algorithms is suggested to deploy mini batch training, that combines online and batch training by dividing the training data into small batches and train on these batches instead of only one large batch.

VIII. CONCLUSION

An experimental shows the feasible of parallel approach in learning to help the speed up performance of ANN for a large scale of data. A lot of efforts have been made to achieve a higher accuracy but only very few bother about time performance. It can be concluded that there are always exist the opportunity in improving time while preserving accuracy of recognition for ANN. The use of parallel in our case help the learning a little bit faster. However, distributing pattern evenly between computer nodes without randomness of pattern resulted bad training time consuming. The candidate solution might be on other parallel strategies of ensemble neural learning for dynamic load balancing in order to find the best method to reduce time while preserving accuracy focusing in large scale data.

REFERENCES

[1] Park, S. S., Jung, W. G., Shin, Y. G. and Jang, D.-S. Optical character recognition system using bp algorithm. *International Journal of Computer Science and Network Security (IJCSNS)*, 8, 12 (Dec, 2008 2008), 118.

[2] Al-Omari, S. A. K., Sumari, P., Al-Taweel, S. A. and Husain, A. J. A. Digital recognition using neural network. *Journal of Computer Science* 5, (6) 2009), 427-434.

[3] Shrivastava, S. and Singh, M. P. Performance evaluation of feed-forward neural network with soft computing techniques for hand written english alphabets. *Applied Soft Computing*, 11, 1 2010), 1156-1182.

[4] Alsmadi, M. K. S., Omar, K. B. and Noah, S. A. Back propagation algorithm : The best algorithm among the multi-layer perceptron algorithm. *International Journal of Computer Science and Network Security (IJCSNS)*, VOL.9, No.4 (April 2009 2009), 378-383.

[5] Suresh, S., Omkar, S. N. and Mani, V. Parallel implementation of back-propagation algorithm in networks of workstations. *IEEE Transactions on Parallel and Distributed Systems*, 16, 1 2005), 24-34.

[6] Liu, C.-L. and Fujisawa, H. "Classification and learning for character recognition: Comparison of methods and remaining problems". City, 2005.

[7] Feng, Y. and Fan, Y. "Character recognition using parallel bp neural network". City, 2008.

[8] Ganapathy, V. and Liew, K. L. "Handwritten character recognition using multiscale neural network training technique". City, 2008.

[9] Kattan, A. R. M., Abdullah, R. and Salam, R. A. "Training feed-forward neural networks using a parallel genetic algorithm with the best must survive strategy". IEEE, City, 2010.

[10] Ganeshamoorty, K. and Ranasinghe, D. N. On the performance of parallel neural network implementations on distributed memory architectures. *Eight IEEE International Symposium on Clster Computing and the Grid*(2008 2008), 90-97.

[11] Enachesu, C. and Miron, C.-D. "Handwritten digits recogntion using neural computing". In *Proceedings of the International Conference Interdiciplinarity in Engineering* (Romania, 2010). INFORMATICA,

[12] Wang, W., Zhao, X. and Feng, X. Parallel wavelet-based image segmentation using mpi. *IEEE*2004).

[13] Kostopoulos, A. E. and Grapsa, T. N. Self-scaled conjugate gradient training algorithms. *Neurocomputing*, 72, 13-15 2009), 3000-3019.

[14] Schrauwen, B., Verstraeten, D. and Campenhout, J. V. "An overview of reservoir computing: Theory, applications and implementations ". In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'2007)*(Bruges, Belgium, 25-27 April 2007, 2007),

[15] Roux, J. L. and McDermott, E. "Optimization methods for discriminative training". City, 2005.

[16] Yu, H. and Wilamowski, B. M. "Efficient and reliable training of neural networks". IEEE Press, City, 2009.

[17] Wena, U.-P., Lan, K.-M. and Shih, H.-S. A review of hopfield neural networks for solving mathematical programming problems. *European Journal of Operational Research*, 198, 3 (1 November 2009 2009), 675–687.

[18] Kalteh, A. M., Hjorth, P. and Berndtsson, R. Review of the self-organizing map (som) approach in water resources: Analysis, modelling and application. *Environmental Modelling & Software*, 23, 7 2008), 835-845.

[19] Perwej, Y. and Chaturvedi, A. Machine recognition of hand written characters using neural networks. *International Journal of Computer Applications*, 14, 2 (January 2011 2011), 6-9.

[20] Grubb, A. and Bagnell, J. A. "Boosted backpropagation learning for training deep modular networks". In *Proceedings of the International Conference on Machine Learning (27th ICML)* (Haifa, Israel, 2010),

[21] Haykin, S. "Neural networks: A comprehensive foundation". Prentice Hall, USA, 1999.

[22] Turchenko, V. and Grandinetti, L. "Efficiency analysis of parallel batch pattern nn training on general purpose supercomputer". Springer Berlin Heiderberg, City, 2009.

[23] Negnevitsky, M. "Artificial intelligence: A guide to intelligent systems". Pearson Education, 2002.

[24] Turchenko, V. Fine grain approach to development of parallel training algorithm of multi-layer perceptron. *Scientific Theoritical Journal of Artificial Intelligence*, 1, UDC 681.3 2006), 94-102.

[25] Llano, R. M. d. and Bosque, J. L. Study of neural net training methods in parallel and distributed architectures. *Elsevier : Future Generation Computer Systems*2009), 267-275.

[26] Dahl, G., McAvinney, A. and Newhall, T. "Parallelizing neural network training for cluster systems". ACTA Press, City, 2008.

[27] Omatu, S., Rocha, M., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J., Turchenko, V. and Grandinetti, L. "Efficiency analysis of parallel batch pattern nn training algorithm on general-purpose supercomputer". Springer Berlin / Heidelberg, City, 2009.

[28] Lotric, U. and Dobnikar, A. "Parallel implementations of feed-forward neural network using mpi and c# on .Net platform". City, 2005.

[29] Pethick, M., Liddle, M., Weerstein, P. and Huang, Z. "Parallelization of a backpropagation neural network on a cluster computer". ACTA Press, City, 2003.

[30] Babii, S., Cretu, V. and Petriu, E. M. "Performance evaluation of two distributed backpropagation

implementations". In *Proceedings of the International Joint Conference on Neural Network (IJCNN 2007)* (Orlando, Florida, USA, 12-17 Aug. 2007, 2007).

[31] Avci, E., Sengur, A. and Hanbay, D. An optimum feature extraction method for texture classification. *Expert Systems with Applications*2009), 6036-6043.

[32] Anthony, T. C. and Jagannathan, S. "A distributed discrete-time neural network architecture for pattern allocation and control". IEEE Computer Society, City, 2002.

[33] Nakama, T. Theoretical analysis of batch and on-line training for gradient descent learning in neural networks. *Neurocomputing*, 73, 1-3 2009), 151-159.

[34] Zhang, W., Tong, W., Chen, Z., Glowinski, R., Liu, Y., Li, Y., Zhang, B. and Wu, G. "Design & implementation of the parallel-distributed neural network ensemble". Springer Berlin Heidelberg, City, 2005.

[35] Gropp, W., Lusk, E. and Skjelmm, A. Using mpi portable parallel programming with the message-passing interface, MIT Press, Cambridge, USA, 1999.

[36] Mohamad, M., Saman, M. Y. M. and Hitam, M. S. "Parallel pattern back propagation neural network training of multilayer perceptron ". In *Proceedings of the First National Doctoral Seminar on Artificial Intelligence Technology* (UKM Bangi, 2010). UKM.